
Bayesian Nonparametric Kernel-Learning

Junier Oliva*, Avinava Dubey*, Barnabás Póczos, Jeff Schneider, Eric P. Xing

Machine Learning Department

Carnegie Mellon University

Pittsburgh, PA 15213

{joliva, akdubey, bapocz, schneide, epxing}@cs.cmu.edu

Abstract

Kernel methods are ubiquitous tools in machine learning. They have proven to be effective in many domains and tasks. Yet, kernel methods often require the user to select a predefined kernel to build an estimator with. However, there is often little reason for the a priori selection of a kernel. Even if a universal approximating kernel is selected, the quality of the finite sample estimator may be greatly effected by the choice of kernel. Furthermore, when directly applying kernel methods, one typically needs to compute a $N \times N$ Gram matrix of pairwise kernel evaluations to work with a dataset of N instances. The computation of this Gram matrix precludes the direct application of kernel methods on large datasets. In this paper we introduce Bayesian nonparametric kernel (BaNK) learning, a generic, data-driven framework for scalable learning of kernels. We show that this framework can be used for performing both regression and classification tasks and scale to large datasets. Furthermore, we show that BaNK outperforms several other scalable approaches for kernel learning on a variety of real world datasets.

1 Introduction

Kernel methods have become a staple approach to modern machine learning. They, along with the representer theorem, have given a tractable way to optimize a myriad of machine learning tasks over broad function classes. Indeed, kernel methods like support vector machines (SVMs), kernel-ridge regression, kernel-PCA, and many others have been shown to be effective on a wide range of domains and applications.

However, despite being a standard and well-studied tool in machine learning one commonly overlooked aspect to kernel methods is the choice of kernel to use. Typically, one is forced to make a specific choice of kernel function to use a priori; once fixed, the choice of kernel will induce a reproducing kernel Hilbert space (RKHS) that optimization occurs over. Even when cross-validating kernel parameters, the function class is limited to a RKHS induced by the often arbitrary choice of the family of kernels being cross-validated. Yet, the choice of kernel can greatly impact performance. Indeed, even when using an universal approximating kernel, the finite sample estimate will be affected by the kernel choice. Given that the choice of kernel is an important free parameter in kernel methods, and generally there is little a priori reasons for kernel selections, a principled and data-driven method for learning kernels is extremely useful.

Moreover, another drawback to kernel methods is that they often do not scale to datasets with a large number of instances. This is because kernel methods typically require the computation of a large Gram matrix of kernel evaluations for all pairs of instances in a dataset. That is, when optimizing over a dataset of N instances using a kernel K a Gram matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ needs to be computed, where $\mathbf{K}_{ij} = K(x_i, x_j)$ and x_i 's are input covariates. When N is in the many thousands or more, the computation of \mathbf{K} will be prohibitive. Furthermore, kernel methods will often require

*These two authors had equal contribution.

manipulations of \mathbf{K} such as taking inverses, which will result in a worse time complexity than the $O(N^2)$ time required for computing \mathbf{K} . Considering that modern datasets are only increasing in size, and complicated machine learning tasks require large datasets for achieving a low risk, it is vital to mitigate the high computational cost of kernel methods.

In order to provide a method that scales to large datasets and adaptively learns the kernel to use in a data-driven fashion, this paper presents the *Bayesian nonparametric kernel-learning* (BaNK) framework. BaNK is a novel approach that will use random features to both provide a scalable solution and learn kernels.

Random features have been recently shown to be an effective way to scale kernel methods to large datasets. Roughly speaking, random feature techniques like random kitchen sinks (RKS) [18] work as follows. Given a shift invariant kernel $K(x, x') = k(x - x')$, one constructs an approximate primal space to estimate kernel evaluations $K(x, x')$ as the dot product of finite vectors $\varphi(x)^T \varphi(x')$. The vectors φ are constructed with random frequencies drawn from a distribution \mathcal{D} that is defined by K . Similarly, a distribution \mathcal{D} from which random frequencies are drawn from defines a kernel K that the random frequencies approximate. It is this last observation that is key for the BaNK framework. Whereas a typical use of RKS will consider only a fixed distribution \mathcal{D} of random frequencies, via the kernel K , BaNK will allow the distribution \mathcal{D} to vary with the given data, effectively learning the kernel. In particular, BaNK shall vary \mathcal{D} with a graphical model approach where we treat \mathcal{D} as a latent parameter and place a prior on it (Figure 1). The prior on \mathcal{D} , along with the data generation model, will allow one to sample from a posterior over \mathcal{D} in order to learn the corresponding kernel.

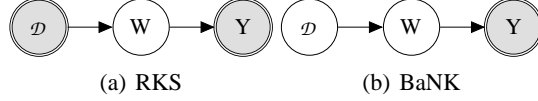


Figure 1: (a) Traditional random feature approach where the distribution \mathcal{D} of random features W is held fixed. (b) BaNK framework where \mathcal{D} is taken to be random.

Modeling \mathcal{D} as a mixture of Gaussians with a Dirichlet process prior allows BaNK to learn a kernel from a rich, broad class. Furthermore, with the use of random features, we are able to efficiently sample the model parameters and work over larger datasets. Moreover, by using Metropolis-Hastings sampling from a proper posterior, the kernels we learn are interpretable and the random features are asymptotically guaranteed to come from the underlying posterior distribution unlike greedy non-convex optimization methods.

Outline The rest of this paper is structured as follows. First we review the use of random features for kernel approximation and show how such an approach can be used for flexible and efficient kernel learning. Second, we detail our graphical model framework both for supervised regression and classification tasks. Third, we expound on our inference method for sampling from the model posterior. Forth, we illustrate the use and performance of BaNK for both regression and classification on several datasets. Lastly we cover related works and give concluding remarks.

2 Model

2.1 Random Features for Kernel Estimation

Below we briefly review the method of random Fourier features for the approximation of kernels [18]. The details of the method will help motivate and explain our BaNK model. Henceforth, we will only consider continuous shift-invariant kernels defined over \mathbb{R}^d : $K(x, y) = k(x - y)$ where $x, y \in \mathbb{R}^d$ and k is a positive definite function. While useful and innovative, the use of random Fourier features for kernel approximation is just a Monte Carlo integration using Bochners theorem [20]. Bochners theorem states that a continuous shift-invariant kernel $K(x, y) = k(x - y)$ is a positive definite function if and only if $k(t)$ is the Fourier transform of a non-negative measure $\rho(\omega)$. Note further, that if $k(0) = 1$, then $\rho(\omega)$ will be a normalized density. That is, if we define $\zeta_\omega(x) \equiv \exp(i\omega^T x)$, then

$$k(x - y) = \int_{\mathbb{R}^d} \rho(\omega) \exp(i\omega^T(x - y)) d\omega = \mathbb{E}_{\omega \sim \rho}[\zeta_\omega(x)\zeta_\omega(y)^*]. \quad (1)$$

Hence, using Monte Carlo integration, we can approximate $K(x, y) = k(x - y)$ using $\omega_j \stackrel{iid}{\sim} \rho$:

$$k(x - y) \approx \frac{1}{M} \sum_{j=1}^M \zeta_{\omega_j}(x) \zeta_{\omega_j}(y)^*. \quad (2)$$

In particular, if our kernel k is real-valued, then we can discard the imaginary part of (2):

$$k(x - y) \approx \varphi(x)^T \varphi(y), \quad \varphi(x) \equiv \frac{1}{\sqrt{M}} [\cos(\omega_1^T x), \dots, \cos(\omega_M^T x), \sin(\omega_1^T x), \dots, \sin(\omega_M^T x)]^T. \quad (3)$$

The great advantage of such an approximation is that we may now estimate a function in the RKHS as a linear operator in the random features: $f(x) = \sum_{i=1}^m \alpha_i K(x_i, x) \approx \sum_{i=1}^m \alpha_i \varphi(x_i)^T \varphi(x) = \psi^T \varphi(x)$, where $\psi = \sum_{i=1}^m \alpha_i \varphi(x_i)$. Thus we may work directly in a primal space of $\varphi(x)$ and avoid computing large Gram matrices. To recap, using the approximation of kernels with random features works as follows: choose a kernel defined by k (with $k(0) = 1$), take its Fourier transform, $p(\omega)$, which will be a pdf over \mathbb{R}^d ; draw M i.i.d. samples from $p(\omega)$, $\{\omega_j\}_{j=1}^M$; estimate the kernel with $K(x, y) \approx \varphi(x)^T \varphi(y)$ as in (3).

However, Bochner's theorem also allows one to work in the other direction. That is, we may start with a distribution \mathcal{D} with pdf $\rho(\omega)$ and take the characteristic function (the inverse Fourier transformation) to define a shift-invariant kernel k . For example, suppose that $\rho(\omega) = \mathcal{N}(\omega|\mu, \Sigma)$, where $\mathcal{N}(\omega|\mu, \Sigma)$ is the pdf of $\mathcal{N}(\mu, \Sigma)$. Taking its characteristic function we see that $k(t) = \exp(i\mu^T t - \frac{1}{2}t^T \Sigma t)$ would be the corresponding shift-invariant kernel. From the kernel learning perspective, Bochner's theorem yields an object to manipulate for the learning of one's kernel: $\rho(\omega)$ the distribution of random features.

We consider distributions that are mixtures of Gaussians:

$$\rho(\omega) = \sum_{k=1}^K \pi_k \mathcal{N}(\omega|\mu_k, \Sigma_k) \rightarrow k(t) = \sum_{k=1}^K \pi_k \exp(i\mu_k^T t - \frac{1}{2}t^T \Sigma_k t). \quad (4)$$

This makes for very general kernels. In fact, i) noting that Gaussian mixture models are universal approximators of densities and may hence approximate any spectral distribution, and ii) using Plancherel's Theorem to relate spectral accuracies to the original domain [23, 28] it follows that:

Proposition 2.1. *The expression of $\rho(\omega)$ in (4) can approximate any shift invariant kernel.*

For the applications we consider we only need real-valued kernels, hence we approximate the real part of (4):

$$K(x, y) = \sum_{k=1}^K \pi_k \exp(-\frac{1}{2}(x - y)^T \Sigma_k (x - y)) \cos(i\mu_k^T (x - y)) \approx \varphi(x)^T \varphi(y), \quad (5)$$

where again $\varphi(x) \equiv \frac{1}{\sqrt{M}} [\cos(\omega_1^T x), \dots, \cos(\omega_M^T x), \sin(\omega_1^T x), \dots, \sin(\omega_M^T x)]^T$ with $\omega_j \stackrel{iid}{\sim} \rho$. An application of the random feature approximation bounds found in [18, 14, 25] yields that:

Proposition 2.2. *For compact $\mathcal{X} \subset \mathbb{R}^d$ with finite diameter, we have that*

$$\Pr \left[\sup_{x, y \in \mathcal{X}} |K(x, y) - \varphi(x)^T \varphi(y)| \geq \epsilon \right] = O \left(\frac{1}{\epsilon^2} \exp \left(\frac{-M\epsilon^2}{4(d+2)} \right) \right).$$

Using the above, it may be seen that one can effectively approximate shift-invariant kernels using random features drawn from Gaussian mixtures. However, in order to learn the kernel, one still needs a mechanism to determine the Gaussian mixture to use. We take a graphical model approach to determine the mixture for $\rho(\omega)$ in a principled, data-driven fashion. The concept of using non-parametric mixture prior on ρ with random frequencies was considered in [26] without empirical details.

2.2 Graphical Model

Graphical models have served as important and effective tool in the learning of latent parameters for a multitude of applications [11]. Hence, we find graphical models to be an incredibly natural tool for

kernel learning. As described above, one may vary and tune kernels with the choice of density over random features, $\rho(\omega)$. Thus, in our model, we take this distribution itself to be a random, latent parameter. It is interesting to note that although it is often restrictive to set a generative process for one's data, the BaNK model, with a stochastic random frequency distribution, is strictly more general than the traditional kernel method approach of using a fixed RBF kernel. Roughly speaking, the BaNK model will consist of three major parts: one, a prior for stochastically generating the random feature distribution $\rho(\omega)$; two, a prior for the generation of the parameters of a linear model in the primal space of random features; three, a generative data model with noise to generate labels given input covariates and the rest of the parameters.

As previously mentioned, a robust and flexible choice of $\rho(\omega)$ is a Gaussian mixture model; we generate it as follows. Since the number of modes of $\rho(\omega)$ is not a priori known, we will assume it to be infinite, but given a finite dataset the model will realize only a finite number of Gaussians in the mixture. Hence we model the distribution as $\rho(\omega) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}(\omega | \mu_k, \Sigma_k)$ where $\sum_k \pi_k = 1$ and $\pi_k > 0$. We use a Dirichlet process (DP) prior on the components of our Gaussian mixture. The Dirichlet process is a distribution over discrete probability measures (i.e., atoms), $G = \sum_{k=1}^{\infty} \pi_k \delta_{\pi_k}$, with countably infinite support, where the finite-dimensional marginals are distributed according to a finite Dirichlet distribution [8]. It is parametrized by a base probability measure H , which determines the distribution of the atom locations, and a concentration parameter $\alpha > 0$ that is proportional to the inverse variance of the atom locations. The DP can be used as the distribution over mixing measures in a nonparametric mixture model. While the DP allows for an infinite number of clusters a priori, any finite dataset will be modeled using a finite, but random, number of clusters. We sample the mixture weights from stick breaking prior which produces samples distributed according to a DP. Thus $\pi \sim GEM(\alpha)$ where GEM is the stick breaking prior. We also put a Normal-Inverse-Wishart prior (which acts as the base measure H) on the mean μ_k and variance Σ_k of each of the Gaussian components.

Above we discussed how functions in a kernel's RKHS can be approximated using a linear mapping in the random features, thus we consider models that operate linearly in the random features using a vector $\beta \in \mathbb{R}^{2M}$. As is standard in Bayesian regression and classification models [3], we generate β from a Normal prior, $\beta \sim \mathcal{N}(\mu_\beta, \sigma I)$.

Lastly, we generate our observations given a dataset $X := (x_1, \dots, x_N)^T$ where each $x_i \in \mathbb{R}^d$. For example in regression tasks we have:

$$y = g(x) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon I), \quad (6)$$

where g is approximated to $\beta^T \varphi(x)$ and $\varphi(x)$ is calculated using (3). Thus $y \sim \mathcal{N}(\beta^T \varphi(x), \sigma_\epsilon)$.

The complete generative model is given below and the corresponding plate diagram is shown in Figure 2.

1. Draw the mixture weights π over components of the kernel. $\pi \sim GEM(\alpha)$.
2. Draw the mixture components from Normal-Inverse-Wishart distribution. I.e. draw $\Sigma_k \sim \mathcal{W}^{-1}(\Psi_0, \nu_0)$, and $\mu_k \sim \mathcal{N}(\mu_0, \frac{1}{\kappa_0} \Sigma_k)$ for $k = 1, \dots, \infty$.
3. For each random frequency index $j = 1, \dots, M$
 - (a) Draw the component from which the frequency vector is drawn. $Z_j \sim Mult(\pi)$.
 - (b) Draw the corresponding random frequency vector $W_j \sim \mathcal{N}(\omega | \mu_{Z_j}, \Sigma_{Z_j})$.
4. Draw the weight vector, $\beta \sim \mathcal{N}(\mu_\beta, \sigma I)$.
5. For each data point index $i = 1, \dots, N$
 - (a) Define $\varphi(X_i)$ as in (3).
 - (b) Draw the observation, e.g. for regression: $Y_i \sim \mathcal{N}(\varphi(X_i)^T \beta, \sigma_\epsilon I)$.

We note that the only change when going from regression to classification is in the step 5(b) of the generative procedure. This time we draw Y_i from a sigmoid.

- 5 For each data point index $i = 1, \dots, N$
 - (b) Draw the output binary label $Y_i \sim \sigma(\varphi(X_i)^T \beta)$, where $\sigma(x) = \frac{1}{1 + \exp(-x)}$.

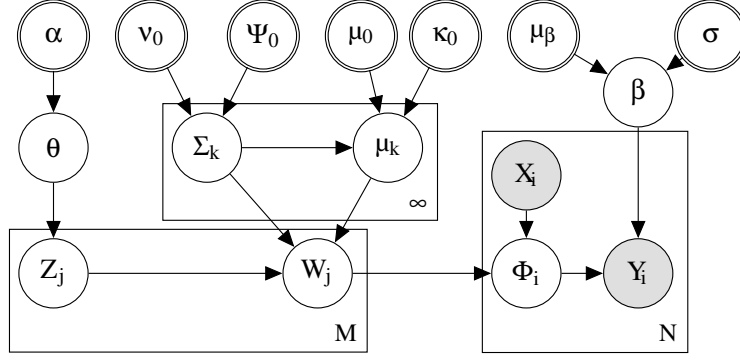


Figure 2: Plate diagram for the graphical model for BaNK learning framework.

3 Inference

We propose a MCMC based solution for inferring the parameters of the mixture of Gaussian distribution that defines $\rho(\omega)$. This includes finding the component assignment vector Z and the mean and covariance μ_k and Σ_k for each component. We will also sample the random frequencies W while marginalizing other parameters including π and β . The sampling equations for Z , μ_k , Σ_k remain the same for both regression and classification and will differ only when sampling W . We will first describe the part of the solution which is common to both regression and classification, and then describe how to get parameters specific to the two tasks.

We want to sample from $p(Z, \mu, \Sigma, W | X, Y, rest)$, where *rest* are all the hyper-parameter of our model while other parameters including β and π have been integrated out. We use Gibbs sampling and sample each variable at a time given all other variable.

3.1 Sampling Z_j

Recall that Z_j indicates which component the random frequency W_j is drawn from. We use the Chinese restaurant process analogy to integrate out π , the component priors, out. Let $m_k \equiv \sum_l \delta(Z_l = k)$. The sampling equation for Z_j can be derived from [17] and is shown below

$$P(Z_j = k | \mu, \Sigma, W, X, Y, rest) \propto \begin{cases} \frac{m_k^{-j}}{M-1+\alpha} \mathcal{N}(\omega_j | \mu_k, \sigma_k) & m_k^{-j} > 0 \\ \frac{\alpha}{M-1+\alpha} \mathcal{N}(\omega_j | \mu_k, \sigma_k) & m_k^{-j} = 0 \end{cases} \quad (7)$$

where $m_k^{-j} = \sum_{l:l \neq j} \delta(Z_l = k)$ and $W_j = \omega_j$. For a new component ($m_k^{-j} = 0$), we sample the mean and the variance from the Normal-Inverse-Wishart prior.

3.2 Sampling μ_k and Σ_k

Given the component assignment Z and the random frequencies W , the posterior distribution of the covariance of each Gaussian component in the mixture is Inverse-Wishart, ie $\Sigma_k \sim \mathcal{W}^{-1}(\Psi_k, \nu_k)$ where $\Psi_k = \Psi_0 + \sum_{j:Z_j=k}^M (W_j - \bar{W}^k)(W_j - \bar{W}^k)^T + \frac{\kappa_0 m_k}{\kappa_0 + m_k} (\bar{W}^k - \mu_0)(\bar{W}^k - \mu_0)^T$, where $\bar{W}^k = \frac{1}{m_k} \sum_{j:Z_j=k} W_j$ and $\nu_k = \nu_0 + m_k$. Similarly, the posterior distribution of μ_k given, Σ_k , Z and W is a normal; i.e. $\mu_k \sim \mathcal{N}(\mu_k, \frac{1}{\kappa_k} \Sigma_k)$, where $\mu_k = \frac{\kappa_0 \mu_0 + m_k \bar{W}^k}{\kappa_0 + m_k}$ and $\kappa_n = \kappa_0 + m_k$. See [10] for more details.

3.3 Sampling W

We derive a Metropolis-Hasting (MH) sampler for sampling W . The posterior distribution of the random frequencies W given the assignment Z , the parameters of the component μ and Σ , and the data, X and Y is proportional to

$$P(W | Z, \mu, \Sigma, Y, X, rest) \propto P(W | Z, \mu, \Sigma) P(Y | X, W, rest). \quad (8)$$

The first term in the LHS is a normal distribution $P(W | Z, \mu, \Sigma) = \prod_j \mathcal{N}(W_j | \mu_{Z_j}, \Sigma_{Z_j})$. Since it is difficult to sample directly from the posterior, we use MH, where the first factor of the LHS of

(8) is used as a proposal distribution; i.e. $Q(W) = P(W|Z, \mu, \Sigma)$. Now, the acceptance ratio for a newly proposed W^* is given by $r = \min \left\{ 1, \frac{P(Y|X, W^*, rest)}{P(Y|X, W, rest)} \right\}$.

Here the ratio is a ratio of model evidences and is calculated differently for regression and classification. Note that the choice to marginalize out β in this manner is intuitive; good random features are those for which there exist a β that models Y well. Hence, we expect a sampling scheme that marginalizes β to be more efficient than one that does not (and instead samples W w.r.t. a current sample of β).

3.3.1 Regression

For regression we make use for conjugacy between prior for β , σ_ϵ and the likelihood to get a closed form solution for $P(Y|X, W, rest)$. In this case we sample σ_ϵ from *Inverse - Gamma*(a_0, b_0). The model evidence is then:

$$P(Y|X, W, rest) = \int_{\beta, \sigma_\epsilon} P(Y|W, X, \beta, \sigma_\epsilon) P(\beta) P(\sigma_\epsilon) d\beta d\sigma_\epsilon \propto \frac{\Gamma(a_n)}{\Gamma(a_0)} \frac{b_0^{a_0}}{b_n^{a_n}} \sqrt{\frac{|\Lambda_0|}{|\Lambda_n|}}, \quad (9)$$

where $\Lambda_0 = \frac{1}{\sigma^2} I$, $\Phi(X) = (\varphi(X_1)^T \dots \varphi(X_N)^T)^T$, $\Lambda_n = \Phi(X)^T \Phi(X) + \Lambda_0$, $\mu_n = \Lambda_n^{-1} (\Lambda_0 \mu_\beta + \Phi(X)^T Y)$, $a_n = a_0 + \frac{n}{2}$ and $b_n = b_0 + \frac{1}{2} (Y^T Y + \mu_0^T \Lambda_0 \mu_0 - \mu_n^T \Lambda_n \mu_n)$. For more details refer to [16]. It is worth noting that one may efficiently compute ratio's of model evidences if proposing a single W_j at a time; this is because doing so amounts to making low-rank updates on $\Phi(X)^T \Phi(X)$.

3.3.2 Classification

The absence of conjugacy makes it difficult to directly estimate the model evidence $P(Y|X, W, rest)$, so instead we estimate the ratio of model evidences. The ratio is computed by approximating the ratios of partition functions (see [4] chapter 11.6 pg 554 for details):

$$r \approx \min \left\{ 1, \frac{1}{L} \sum_{l=1}^L \frac{P(Y|X, W^*, \beta_l)}{P(Y|X, W, \beta_l)} \right\}, \quad (10)$$

where $\beta_l \sim P(\beta|Y, X, W)$ $l = 1, \dots, L$. Since the Gaussian prior is not conjugate to sigmoid, we approximate the posterior $P(\beta|Y, X, W)$ with a Gaussian (see [24] for details on quadratic approximation). The mean (β_0) of the Gaussian is the mode of the posterior $P(\beta|Y, X, W)$, which can be calculated using numerical optimization. By using Laplace approximation of the posterior, the inverse of the covariance can be found as $S_n = -\nabla^2 \log P(\beta|Y, W, X, rest)|_{\beta=\beta_0}$.¹

4 Experiments

We illustrate the use and performance of BaNK for both regression and classification on synthetic and real-world datasets below.

4.1 Synthetic Data

We give a simple 1-d kernel learning illustration with BaNK using synthetic data. We consider the shift-invariant kernel $k(t) = \exp\left(-\frac{1}{2(2^2)}t^2\right) \left(\frac{1}{2} + \frac{1}{2} \cos(\frac{3}{4}\pi t)\right)$; that is, the kernel whose random frequency distribution is $\rho(\omega) = \frac{1}{2}\mathcal{N}(\omega|0, \frac{1}{2^2}) + \frac{1}{2}\mathcal{N}(\omega|\frac{3}{4}\pi, \frac{1}{2^2})$ (see Figure 3). We look to learn the underlying kernel using 250 frequencies. We generated $N = 1000$ instances $D = \{X_i, Y_i\}_{i=1}^N$ where $X_i \stackrel{iid}{\sim} \mathcal{N}(0, 4^2)$, $Y_i \sim \mathcal{N}(\varphi_\rho(X_i)^T \beta, 1)$, with φ_ρ being the random features from the kernel's true spectral distribution $\omega_j \stackrel{iid}{\sim} \rho$ and

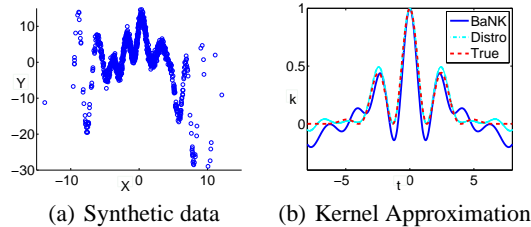


Figure 3: (a) Synthetic data used. (b) True k in dashed red, k estimated with true spectral distribution ρ in cyan, and BaNK estimate in blue.

¹See <http://www.cedar.buffalo.edu/~srihari/CSE574/Chap4/Chap4-Part5.pdf> for details of derivation

$\beta \sim \mathcal{N}(0, I)$. As explained above, using

BaNK one may estimate ρ by drawing from the posterior. We plot such a draw in Figure 3(b). One can see that even though the underlying spectral distribution is multi-model, and the kernel is not easily decernable to the human eye based on the data plot (Figure 3(a)), BaNK approximates the kernel rather well.

Table 1: Regression MSE on UCI MLR.

Dataset	N	d	RKS	BaNK	AlaC	MKL
concrete	1030	8	0.1295 ± 0.0088	0.1204 ± 0.0094	0.2107 ± 0.0592	0.0978 ± 0.0079
noise	1503	5	0.6945 ± 0.0116	0.5013 ± 0.1107	0.7818 ± 0.0689	0.2936 ± 0.2936
prop	11934	16	0.0069 ± 0.0005	$2.4 \times 10^{-5} \pm 1.8 \times 10^{-6}$	0.0019 ± 0.0002	$0.0001 \pm 6.4 \times 10^{-6}$
bike	17379	12	0.2056 ± 0.0034	0.0589 ± 0.0021	0.0648 ± 0.0044	0.0935 ± 0.0026
tom's	28179	96	0.0796 ± 0.0344	0.0153 ± 0.0061	0.0811 ± 0.0308	0.1007 ± 0.0320
music	515345	90	0.8488 ± 0.0027	0.7386 ± 0.0041	0.6827 ± 0.0033	0.8078 ± 0.0017
twitter	583250	77	0.3819 ± 0.0511	0.0777 ± 0.0214	0.3168 ± 0.0877	0.4333 ± 0.0588

4.2 Regression

Below we run experiments with various real-world datasets found in the UCI machine learning repository (UCI MLR)². We compare BaNK to the straight-forward random feature approach with a fixed kernel as well as other competitive random feature based kernel learning methods. In particular we compare to the following methods:

RKS For this method we take input covariates to be random features $\varphi(x_i)$ as in (3). Here we take the random frequencies $\omega_j \stackrel{iid}{\sim} \mathcal{N}(0, \sigma^{-2}I)$. This corresponds to approximating the RBF kernel: $K(x_i, x_l) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_l\|^2\right)$.

MKL One of the most widely used approaches to kernel-learning is multiple kernel learning (MKL) [1, 13]. Here, one attempts to learn a kernel using a non-negative linear combination of a fixed bank of kernels. That is, MKL attempts to learn a kernel K :

$$K(x_i, x_l) = \sum_{m=1}^M \alpha_m K_m(x_i, x_l), \text{ where } \alpha_m \geq 0, \quad (11)$$

and K_1, \dots, K_M are predefined kernels. The kernel weights α_m would then be optimized according to one's loss. Note that (11) still requires the computation of a $N \times N$ Gram matrix, in fact, it requires M such Gram matrices. However, we extend MKL to use random features and scale to larger datasets. If $K_m(x_i, x_l) \approx \varphi_m(x_i)^T \varphi_m(x_l)$, then

$$K(x_i, x_l) \approx \sum_{m=1}^M \alpha_m \varphi_m(x_i)^T \varphi_m(x_l) = \bar{\varphi}(x_i)^T \bar{\varphi}(x_l), \quad (12)$$

where $\bar{\varphi}(x_i) = [\sqrt{\alpha_1}\varphi_1(x_i)^T, \dots, \sqrt{\alpha_M}\varphi_M(x_i)^T]^T$. Hence, it is possible to work directly over input covariates of $\bar{\varphi}(x_i) = [\varphi_1(x_i)^T, \dots, \varphi_M(x_i)^T]^T$, the concatenation of the random features for each kernel K_1, \dots, K_M . We take our bank of kernels to be Laplace, RBF, and Cauchy kernels at various scalings.

AlaC Very recently, independent work by [28] has considered an optimization approach, called Ala Carte, to learning a mixture of kernels. Here, an unconstrained, unpenalized, and non-convex problem is posed for regression and optimized over the parameters of the mixture model and linear weights. Specifically, we optimized the following problem based on [28]³:

$$\min_{\beta, \nu, M, \mu} \frac{1}{2} \sum_{i=1}^N \left\{ Y_i - \sum_{k=1}^K \nu_k^2 \sum_{j=1}^D \left(\beta_{kj}^{\cos} \cos(X_i^T M_k w_{kj} + X_i^T \mu_k) + \beta_{kj}^{\sin} \sin(X_i^T M_k w_{kj} + X_i^T \mu_k) \right) \right\}^2 + \frac{\lambda}{2} (\|\beta^{\cos}\|^2 + \|\beta^{\sin}\|^2),$$

²<https://archive.ics.uci.edu/ml/index.html>

³We note that [28] uses approximate Gaussian matrices in a method called Fastfood [14]. However, we found no increase in speed for the matrices of sizes considered in the experiments (using the implementation found in <http://www.mathworks.com/matlabcentral/fileexchange/49142-fastfood-kernel-expansions>), so we draw w_{kj} from a Gaussian.

where $\beta^{\cos} \in \mathbb{R}^{KD}$, $\beta^{\sin} \in \mathbb{R}^{KD}$, $\nu \in \mathbb{R}^K$, $M_k \in \mathbb{R}^{d \times d}$, $\mu_k \in \mathbb{R}^d$ are optimized; $w_{kj} \in \mathbb{R}^d$ are standard Gaussian vectors that are drawn before optimizing and held fixed.

We perform 5-fold cross-validation (picking parameters on validation sets and reporting back the error on test sets). The total number of random features was chosen to be 500. Below we report the mean squared error (MSE) \pm standard errors; for better interpretability, we standardized the output responses. As per [28], we optimized AlaC using LBFGS⁴. We note although the optimization of the AlaC problem above took on average nearly twice as long as our sampling, we achieved a lower MSE on a majority of the datasets.

4.3 Classification

As previously mentioned, we may use the BaNK framework to perform kernel learning in classification tasks. Below we illustrate the use of BaNK for classification and kernel learning on real-world datasets from the UCI MLR. Furthermore, we compare the accuracies of the models found with BaNK to traditional scalable kernel methods for classification namely RKS and MKL using a logistic model. We see that BaNK proves to be an effective method in classification as well.

Table 2: Classification Prediction Error on UCI MLR.

Dataset	N	d	RKS	MKL	BaNK
PIMA	768	8	0.2571 ± 0.0258	0.2468 ± 0.0229	0.2338 ± 0.0148
Diabetic Retinopathy Debrecen	1151	20	0.2638 ± 0.0205	0.3138 ± 0.0257	0.2603 ± 0.0173
Skin segmentation	2103	3	0.0313 ± 0.0057	0.1706 ± 0.1481	0.0209 ± 0.0028
EEG Eye State	14980	15	0.1322 ± 0.0030	0.1366 ± 0.0764	0.0992 ± 0.0028
Statlog Space Shuttle	58000	9	0.0028 ± 0.0010	0.0018 ± 0.0001	0.0009 ± 0.0001

5 Related Works

Given the poor scaling of kernel methods on datasets with many instances, several methods have looked at approximations of kernels that bypass the computation of large Gram matrices. For example, Nystöm based methods [7] look to give a fast to compute, low-rank approximation of the Gram matrix. Other approaches for summarizing the Gram matrix by the removal of elements or rows have been explored in [21, 5, 9, 22]. Furthermore, approximations based on KD-trees have been explored in [22]. In this paper, we work with kernel approximations based on random features, called Random Kitchen Sinks [18, 19, 14]. As previously discussed, random feature approaches work using an empirical estimate of kernels that stem by drawing features from the Fourier transform of positive definite functions, which will be a distribution if properly scaled.

Kernel learning methods have also received attention due to the impact that kernel choice has on performance. Indeed, even if one fixes a family of kernels to use (e.g. RBF kernels) one still has to select the parameters of the kernels. This is often done with cross-validation or with methods like [12, 6]. A very popular approach to learning kernels in a more flexible class is multiple kernel learning (MKL) [1, 13]. As illustrated above, in MKL, one looks to learn a kernel that is the non-negative linear combination of a bank of fixed kernels. However, naively applying MKL approaches would still require the computation of several large Gram matrices; instead, as previously discussed, one may combine random features to perform scalable MKL (see [15] for an application of such ideas). Very recently, independent work [28] has explored an optimization approach, A la carte, to learning parameters of mixture of kernels, where one optimizes the parameters generating random features in non-convex likelihood problems. See also [2] for another optimization approach. Unfortunately, due to the non-convex nature of the optimization problem being optimized, such approaches yield non-interpretable kernels, whereas BaNK yield draws from a well defined posterior. [27] considers kernels of the form in (4), with parameters chosen heuristically; [26] mentions the possibility of putting a DP prior model on parameters, but does so without a sampling algorithm and empirical details.

6 Conclusion

In this paper we propose an efficient and general data-driven framework, BaNK, for learning of kernels that scales to large datasets. By representing the spectral density using a non-parametric

⁴Using <https://github.com/pcarbo/lbfgsb-matlab>.

mixture of Gaussians, we capture a large class of kernels that can be learned. We provide a generative model for learning kernels while performing regression and classification tasks, and propose novel MCMC based sampling schemes to infer parameters of the mixtures. We show that our proposed framework outperforms other scalable kernel learning methods on a variety of real world datasets in both classification and regression task.

References

- [1] F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.
- [2] E. G. Băzăvan, F. Li, and C. Sminchisescu. Fourier kernel learning. In *Computer Vision—ECCV 2012*, pages 459–473. Springer, 2012.
- [3] C. M. Bishop and M. E. Tipping. Bayesian Regression and Classification. *IOS Press*, 2003.
- [4] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.
- [5] A. Blum. Random projection, margins, kernels, and feature-selection. In *Subspace, Latent Structure and Feature Selection*, pages 52–68. Springer, 2006.
- [6] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.
- [7] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.
- [8] T. S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2): 209–230, Mar. 1973.
- [9] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.
- [10] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2003. ISBN 158488388X.
- [11] M. I. Jordan. Graphical models. *Statist. Sci.*, 19:140–155, 02 2004.
- [12] S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient-based adaptation of hyper-parameters in svm models. 2007.
- [13] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.
- [14] Q. Le, T. Sarlos, and A. Smola. Fastfood—approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.
- [15] Z. Lu, A. May, K. Liu, A. B. Garakani, D. Guo, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, et al. How to scale up kernel methods to be as good as deep neural nets. *arXiv preprint arXiv:1411.4000*, 2014.
- [16] T. P. Minka. Bayesian linear regression. Technical report, 3594 Security Ticket Control, 1999.
- [17] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Technical Report 9815, Dept. of Statistics, University of Toronto, 1998.
- [18] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.
- [19] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.
- [20] W. Rudin. *Fourier analysis on groups*. Number 12. John Wiley and Sons, 1990.
- [21] D. A. F. M. B. Scholkopf. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, volume 1, page 335. MIT Press, 2002.
- [22] Y. Shen, A. Ng, and M. Seeger. Fast gaussian process regression using kd-trees. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161316, 2006.
- [23] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.
- [24] D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, 1990.
- [25] D. Sutherland and J. Schneider. On the error of random fourier features. In *AISTATS*, 2015.
- [26] A. G. Wilson. A process over all stationary covariance kernels. Technical report, 2012.

- [27] A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. *ICML*, 2013.
- [28] Z. Yang, A. J. Smola, L. Song, and A. G. Wilson. A la carte-learning fast kernels. In *AISTATS*, 2015.